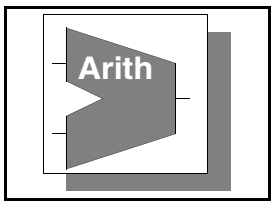
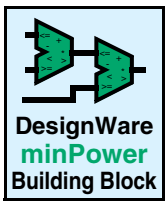




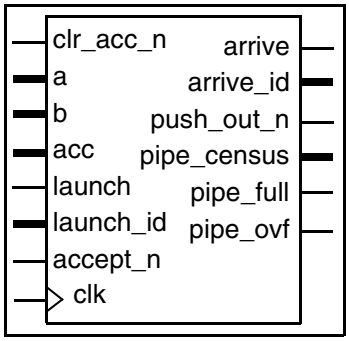
DW_piped_mac

Pipelined Multiplier-Accumulator

Version, STAR, and myDesignWare Subscriptions: IP Directory

Features and Benefits

Revision History



- Integrated multiply and accumulate
- Built-in pipeline and power management
- Parameterized operand widths
- Parameterized multiply and accumulate output width
- Parameterized pipeline stages
- Launch identifier tracking propagation
- DesignWare datapath generator is employed for better timing and area
- Provides minPower benefits (see Table 1-3 on page 3)

Description

The DW_piped_mac performs a multiply and accumulate based on two operands. The operation can be configured to be pipelined. Power management capability is integrated and applied to pipelined stages that are configured into the design.

Table 1-1 Pin Description

Pin Name	Width	Direction	Function
clk	1 bit	Input	Input clock source
rst_n	1 bit	Input	Asynchronous reset (active low)
init_n	1 bit	Input	Synchronous reset (active low)
clr_acc_n	1 bit	Input	Clear accumulator results for upcoming product (active low)
a	*a_width* bits	Input	Multiplier
b	*b_width* bits	Input	Multiplicand
acc	*acc_width* bits	Output	Multiply and accumulate result
launch	1 bit	Input	Indicator to begin a new multiply and accumulate
launch_id	*id_width* bits	Input	Identifier for the corresponding asserted `launch`
pipe_full	1 bit	Output	Upstream notification that pipeline is full
pipe_ovf	1 bit	Output	Status Flag indicating pipe overflow

Table 1-1 Pin Description (Continued)

Pin Name	Width	Direction	Function
accept_n	1 bit	Input	`acc` result accepted from downstream logic (active low)
arrive	1 bit	Output	`acc` result is valid
arrive_id	*id_width* bits	Output	`launch_id` from the originating `launch` that produced the `acc` result
push_out_n	1 bit	Output	Used with external FIFO (optional) to indicate a new `acc` result has been accepted from the pipeline (active low)
pipe_census	3 bits	Output	Output bus indicates the number of pipe stages currently occupied

Table 1-2 Parameter Description

Parameter	Values	Description
a_width	≥ 1 Default: 8	Word length of a
b_width	≥ 1 Default: 8	Word length of b
acc_width	≥ *a_width* + *b_width* Default: 16	Word length of `acc`
tc	0, 1 Default: 0	Two's complement for internal multiply ■ 0: Unsigned ■ 1: Signed
pipe_reg	0 to 7	Pipeline register insertion Insertion of pipeline register stages. For settings and their meanings, see Table 1-5 on page 5.
id_width	1 to 1024 Default: 8	Launch identifier width
no_pm	0, 1 Default: 0	Omit pipe manager For more, see "Omitting Pipe Manager" on page 6

Table 1-2 Parameter Description (Continued)

Parameter	Values	Description
op_iso_mode	0 to 4 Default: 0	Operand isolation mode (controls datapath gating for minPower flow) Allows you to set the style of minPower datapath gating for this module ■ 0: Use the DW_lp_op_iso_mode[a] synthesis setting ■ 1: 'none' ■ 2: 'and' ■ 3: 'or' ■ 4: Preferred gating style: ‘and’ NOTE: Datapath gating is inserted at the inputs of the lpwr implementation only when stage0 is not included (as controlled by the value of *pipe_reg and* described in Table 1-5 on page 5). For details about enabling minPower datapath gating for this component, see “Enabling minPower” on page 13.

a. The DW_lp_op_iso_mode synthesis variable is available only in Design Compiler.
DW_lp_op_iso_mode sets a global style of datapath gating. To use the global style, set *op_iso_mode* to '0', Note that If the *op_iso_mode* parameter is set to '0' and DW_lp_op_iso_mode is either not set or set to 0', then no datapath gating is inserted for this component.

Table 1-3 Synthesis Implementations

Implementation Name	Implementation	License Feature Required
str	Pipelined str synthesis model	DesignWare
lpwr[a]	Low Power synthesis model	■ DesignWare (P-2019.03 and later) ■ DesignWare-LP (before P-2019.03)

a. Requires that you enable minPower; for details, see “Enabling minPower” on page 13.
When minPower is enabled, the lpwr implementation is always chosen during synthesis.

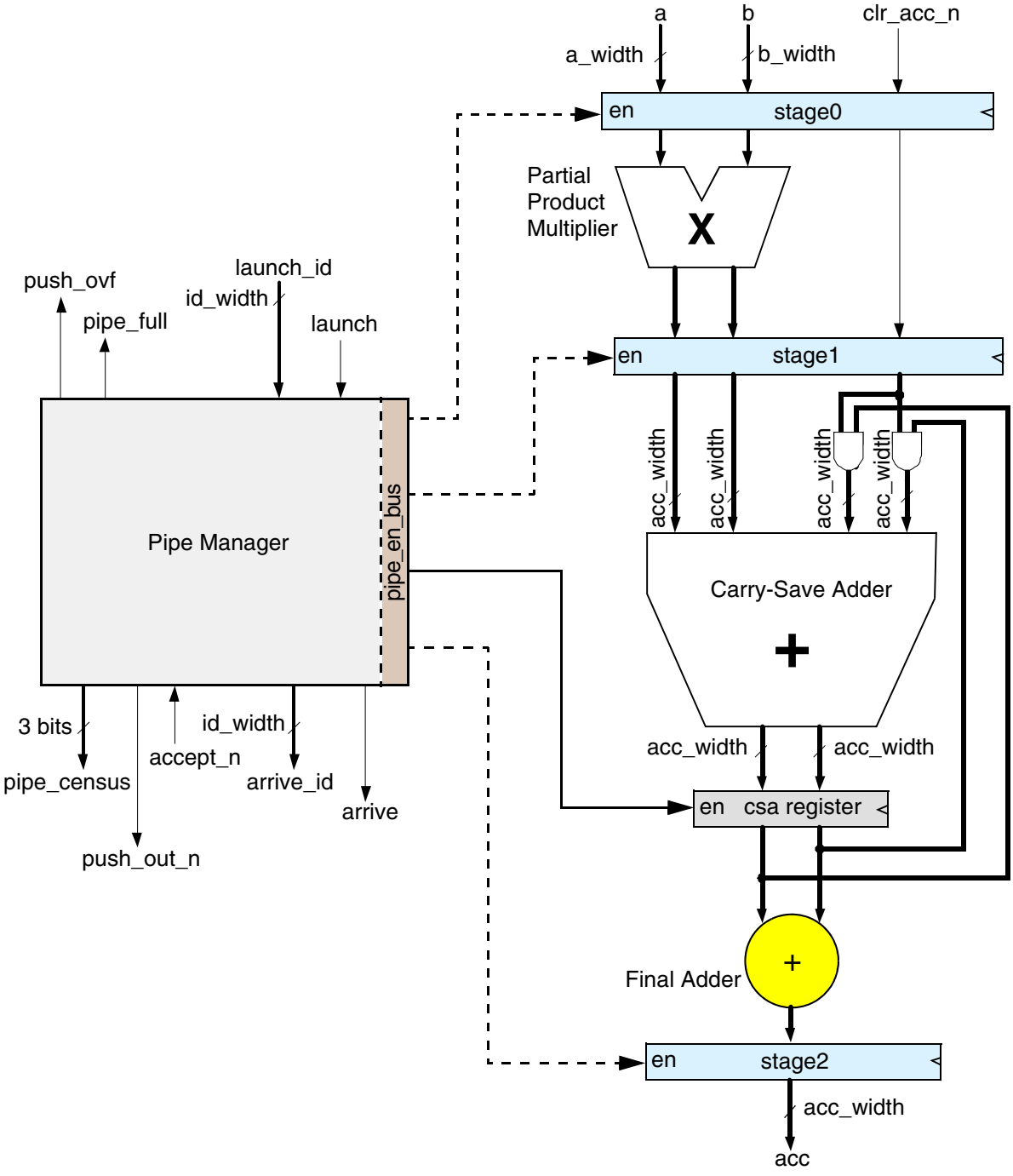
Table 1-4 Simulation Models

Model	Function
DW03.DW_PIPED_MAC _CFG_SIM	Design unit name for VHDL simulation
dw/dw03/src/DW_piped_mac_sim.vhd	VHDL simulation model source code
dw/sim_ver/DW_piped_mac.v	Verilog simulation model source code

Block Diagram

Figure 1-1 on page 4 is a detailed block diagram of the DW_piped_mac.

Figure 1-1 DW_piped_mac Block Diagram



Functional Description

The following table describes values for the pipe_reg parameter.

Table 1-5 pipe_reg Parameter Encodings

pipe_reg Value		Pipeline Register Stages Inserted**
decimal	binary	
0	000	no pipeline stages inserted
1	001	stage0
2	010	stage1
3	011	stage1, stage0
4	100	stage2
5	101	stage2, stage0
6	110	stage2, stage1
7	111	stage2, stage1, stage0

Note: ** - See DW_piped_mac block diagram for pipe register stage references.

Pipelining

The DW_piped_mac is configurable for pipeline register stage insertion. Setting the value for the parameter `pipe_reg` (see Table 1-2 on page 2 and Figure 1-1 on page 4) determines which pipeline register stages are inserted. There will always be at least one register stage called “csa register” that captures the accumulator results (see Figure 1-1 on page 4). Therefore, depending on the parameter `pipe_reg` setting, the number of clock cycles for the `acc` result to propagate ranges from 1 to 4.

Clearing Accumulator Results

An active-low input called `clr_acc_n` is available to clear out the accumulator results for the accompanying operand product. This mechanism provides a method to begin a new series of multiply and accumulate results with a clean slate and minimal latency. The `clr_acc_n` signal is pipelined the same amount as the input operands (and their product) so they arrive at the accumulator in the same clock cycle.

Pipeline Management (Pipe Manager)

Running in parallel to the DW_piped_mac pipeline is a pipeline tracking shift register that monitors the activity. This block is called the Pipe Manager. In cases where there is inactivity on a particular stage of the DW_piped_mac pipeline, the pipeline tracking shift register within Pipe Manager disables those stages to promote power savings. Furthermore, if using the Synopsys Power Compiler tool, the presence of the pipeline tracking shift register and its wiring to the DW_piped_mac pipeline provides an opportunity for increased power reduction in the form of clock gating.

Along with potential power savings that the Pipe Manager provides, it can be used to improve performance in cases where intermittent launch operations are present and there contains first-in first-out (FIFO) structures upstream and downstream of the DW_piped_mac. The handshake is made between the DW_piped_mac and the external FIFOs via the `accept_n` and `pipe_full` ports. Effectively, the DW_piped_mac can considered part of the external FIFO structures. The performance gain comes when inactive (bubbles) stages are detected. These pipeline bubbles are removed to produce a contiguous set of active pipeline stages. The result is empty pipeline slots at the head of (or entering) the DW_piped_mac pipeline for new operations to be launched. Advancing the shifting of operations through the pipeline when a valid `acc` result is available (`arrive` = 1) is controlled by the `accept_n` input. When the DW_piped_mac pipeline is full of active entries, the `pipe_full` output is 1. To disable this feature in cases where no external FIFOs are present, set the `accept_n` input to 0, which effectively eliminates any flow control. At the same time, the `pipe_full` output would always be 0.

To assist in tracking of launched operands, the Pipe Manager provides interface ports call `launch_id` and `arrive_id`. The `launch_id` input is given a value during an active launch operation. Given that `launch_id` values are unique in successive launch operations, the `acc` results can be distinguished from one another with the assertion of `arrive` and the associated `arrive_id`. The `arrive_id` is the `launch_id` from the originating launch that produced the valid `acc` result.

Omitting Pipe Manager

In cases where a meaningful DW_pipe_mac `acc` result is desired every clock cycle, no tracking by identification is needed per `acc` result, and there is no needed to interface with external FIFOs before and after Pipe Manager, the parameter `no_pm` is provided to improve quality of results. When `no_pm` is set to 1, the Pipe Manager is effectively removed from the functionality of DW_piped_mac as well as resulting synthesized netlists. For simulation purposes, the outputs attributed to by the Pipe Manager in DW_piped_mac are tied to fixed values when *no_pm* is set to 1. Table 1-6 shows the values that each Pipe Manager related signal are set to at the DW_piped_mac interface.

Table 1-6 Fixed Outputs When *no_pm* = 1

DW_piped_mac Outputs (from Pipe Manager)	Fixed Value
arrive_id	all 0's
arrive	1
pipe_full	0
pipe_ovf	0
push_out_n	0
pipe_census	all 1's

Suppressing Warning Messages During Verilog Simulation

The Verilog simulation model includes macros that allow you to suppress warning messages during simulation.

To suppress all warning messages for all DWBB components, define the DW_SUPPRESS_WARN macro in either of the following ways:

- Specify the Verilog preprocessing macro in Verilog code:

  ```
  `define DW_SUPPRESS_WARN
  ```

- Or, include a command line option to the simulator, such as:

  `+define+DW_SUPPRESS_WARN` (which is used for the Synopsys VCS simulator)

The warning messages for this model include the following:

- If values other than 1 or 0 are present on a clock port, the following message is displayed:

  ```
  WARNING: <instance_path>.<clock_name>_monitor:
      at time = <timestamp>, Detected unknown value, x, on <clock_name> input.
  ```

  To suppress only this warning message for all DWBB components, use the following macro:

  - Define the DW_DISABLE_CLK_MONITOR macro. You can define this macro in the following ways:
    - Specify the Verilog preprocessing macro in Verilog code:

      ```
      `define DW_DISABLE_CLK_MONITOR
      ```

    - Or, include a command line option to the simulator, such as:

      `+define+DW_DISABLE_CLK_MONITOR` (which is used for the Synopsys VCS simulator)

  This message is also suppressed using the DW_SUPPRESS_WARN macro explained earlier.

Timing Waveforms

Figure 1-2 depicts the general operation of DW_piped_mac with the instantiated component Pipe Manager enabled (*no_pm* = 0). In this configuration, the assertion of `launch` initiates a calculation based on the values driven on `a` and `b`. The `launch_id` input provides a way of tracking the transaction through the pipeline as its results arrive at `acc`. When `launch` is 0 no new calculations are initiated. The pipeline is allowed to advance when the `accept_n` input is asserted (0). A newly calculated `acc` is indicated by `arrive` going to 1. At the same time `arrive` is asserted, the output called `arrive_id` indicates this is the result from the `a` and `b` launched with the `launch_id` at the beginning of the pipeline.

For this configuration, the pipeline is only a single register stage since `pipe_reg` is 0. That means, based on the Block Diagram, the calculated results on `acc` come on the cycle after `a` and `b` are launched.

Another thing to point out in this timing diagram is the assertion of `clr_acc_n` (active low). When `clr_acc_n` is asserted on a `launch` event, the previous `acc` result in DW_piped_mac is cleared and the resulting `acc` will be purely a multiplied by `b`.

Figure 1-2 Launch Every Other Cycle, accept_n Always 0 (*pipe_reg* = 0, *no_pm* = 0)

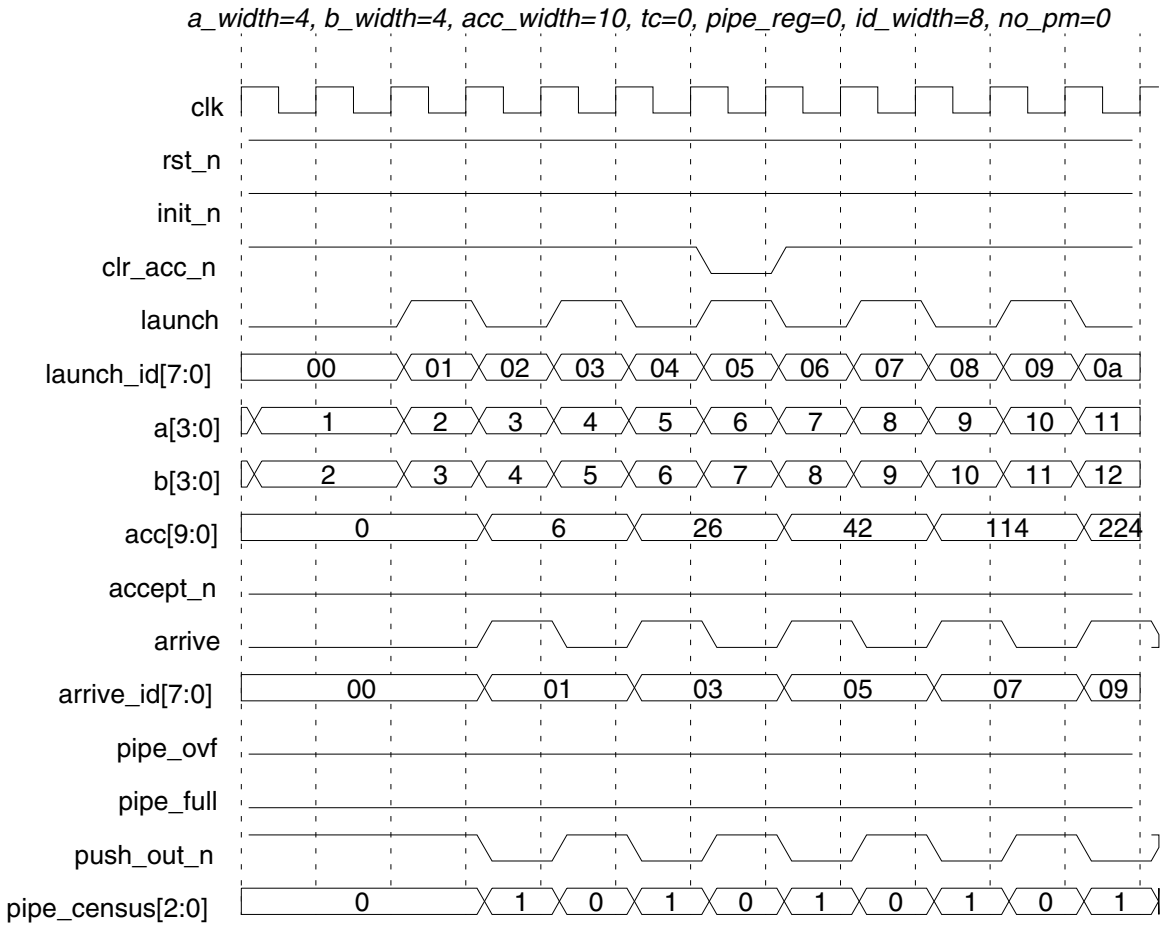


Figure 1-3 depicts a scenario in which `launch` is asserted while `assert_n` is de-asserted until the pipeline of DW_piped_mac becomes full as indicated by `pipe_full` going to 1. Once the pipeline is full, then `launch` is de-asserted and `accept_n` gets asserted some time later to shuffle out the `acc` results and empty the pipeline. In cases where `launch` is asserted after `pipe_full` goes to 1 and while `accept_n` is 1 (de-asserted), a pipeline overflow condition will occur indicated by `pipe_ovf` going to 1. Figure 1-6 on page 12 shows the overflow scenario.

Figure 1-3 Launch Until Pipeline Full, then accept_n Until Empty

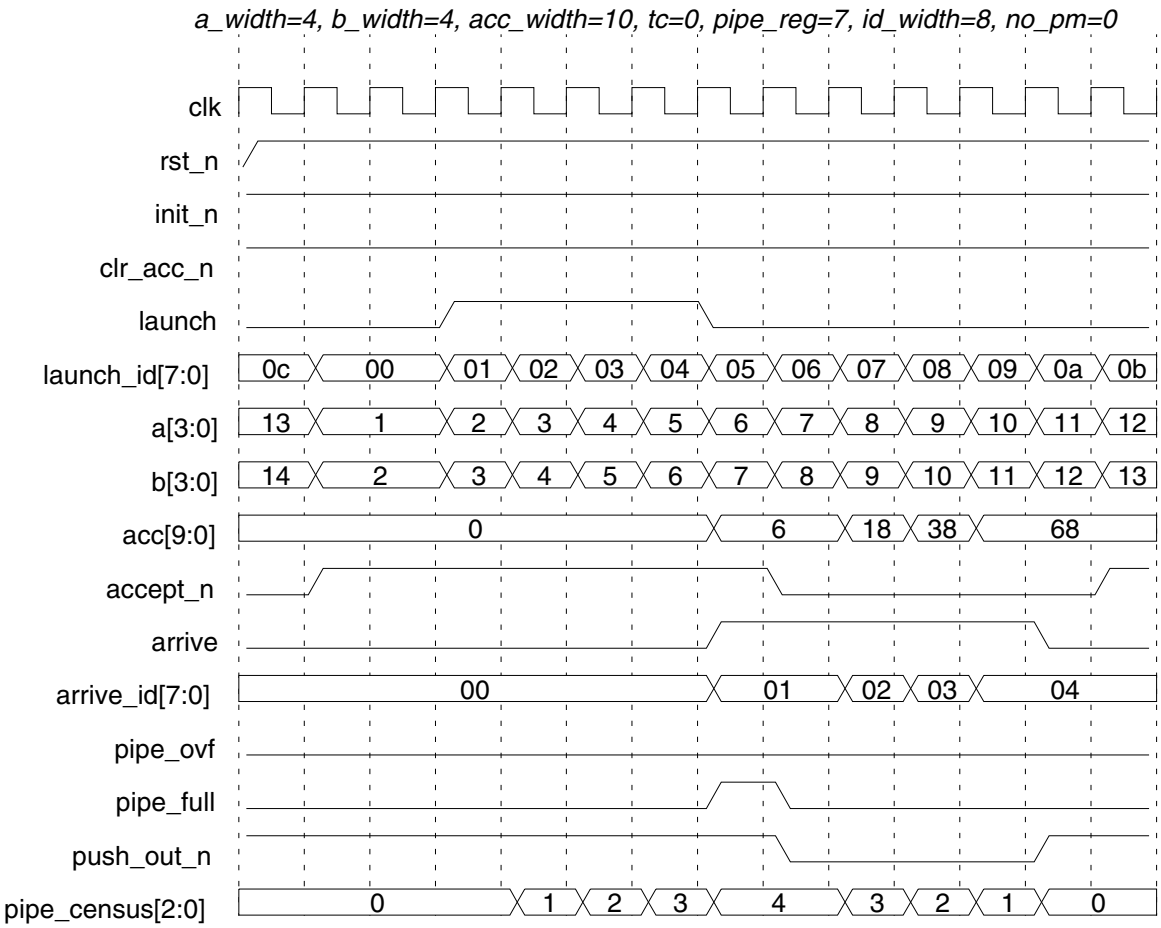


Figure 1-4 depicts a two's complement configuration as it applies to inputs `a` and `b` and output `acc`.

Figure 1-4 Two's Complement Configuration (*tc* = 1, *pipe_reg* = 2, *no_pm* = 0)

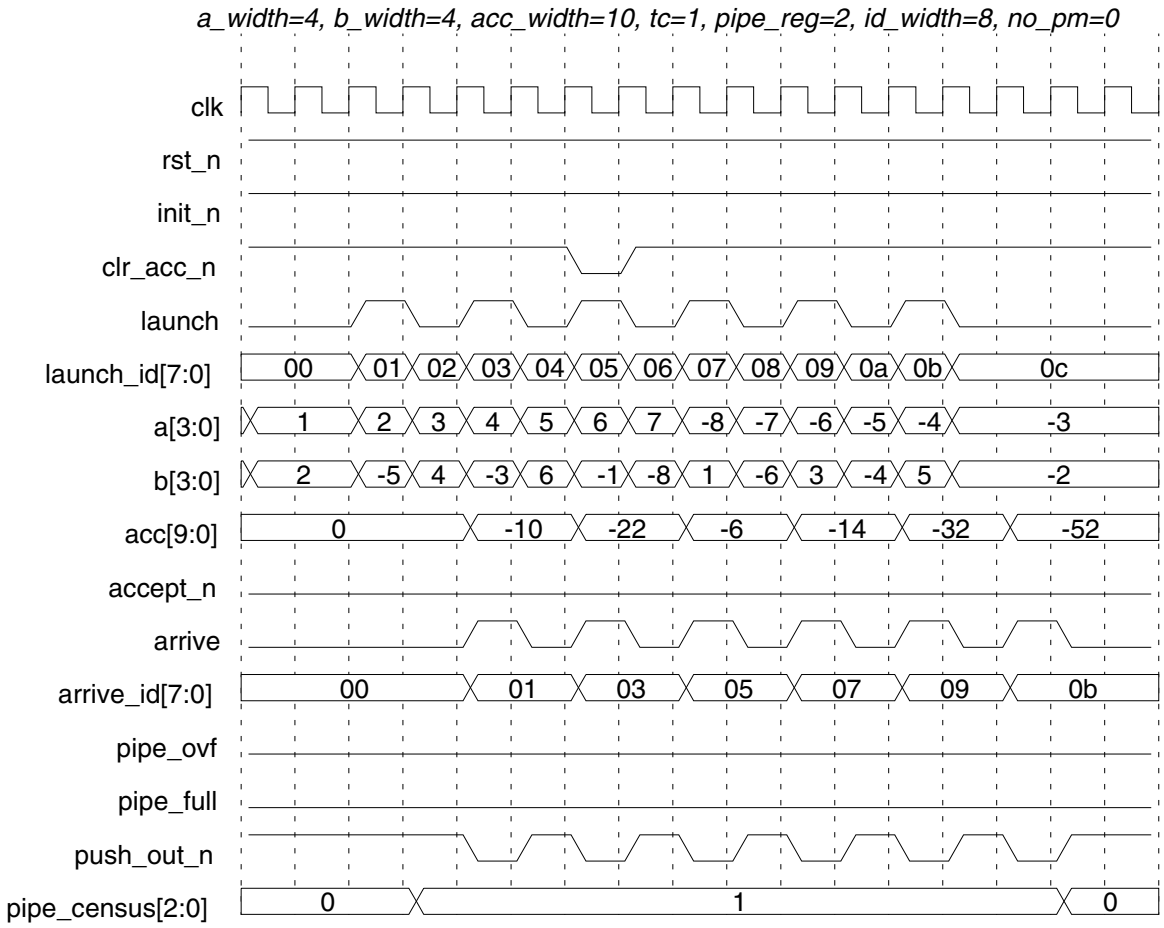


Figure 1-5 depicts a configuration where `no_pm` is 1, which implies that the inputs and outputs related to the Pipe Manager block have no meaning. As the timing diagram shows, even though `launch` is toggling, the results on `acc` are updated every clock cycle just as if `launch` were always 1. Note that all the outputs of DW_piped_mac that are attributed to by Pipe Manager, `arrive`, `arrive_id`, `pipe_ovf`, `pipe_full`, `push_out_n`, and `pipe_census` are held at fixed values. Also note, that `arrive_n` even though not depicted here could be set to 1 and `acc` would still be updating every cycle with the number of pipeline stages inserted as defined by the parameter *pipe_reg*.

Figure 1-5 Configuration Omitting Pipe Manager (*pipe_reg* = 3, *no_pm* = 1)

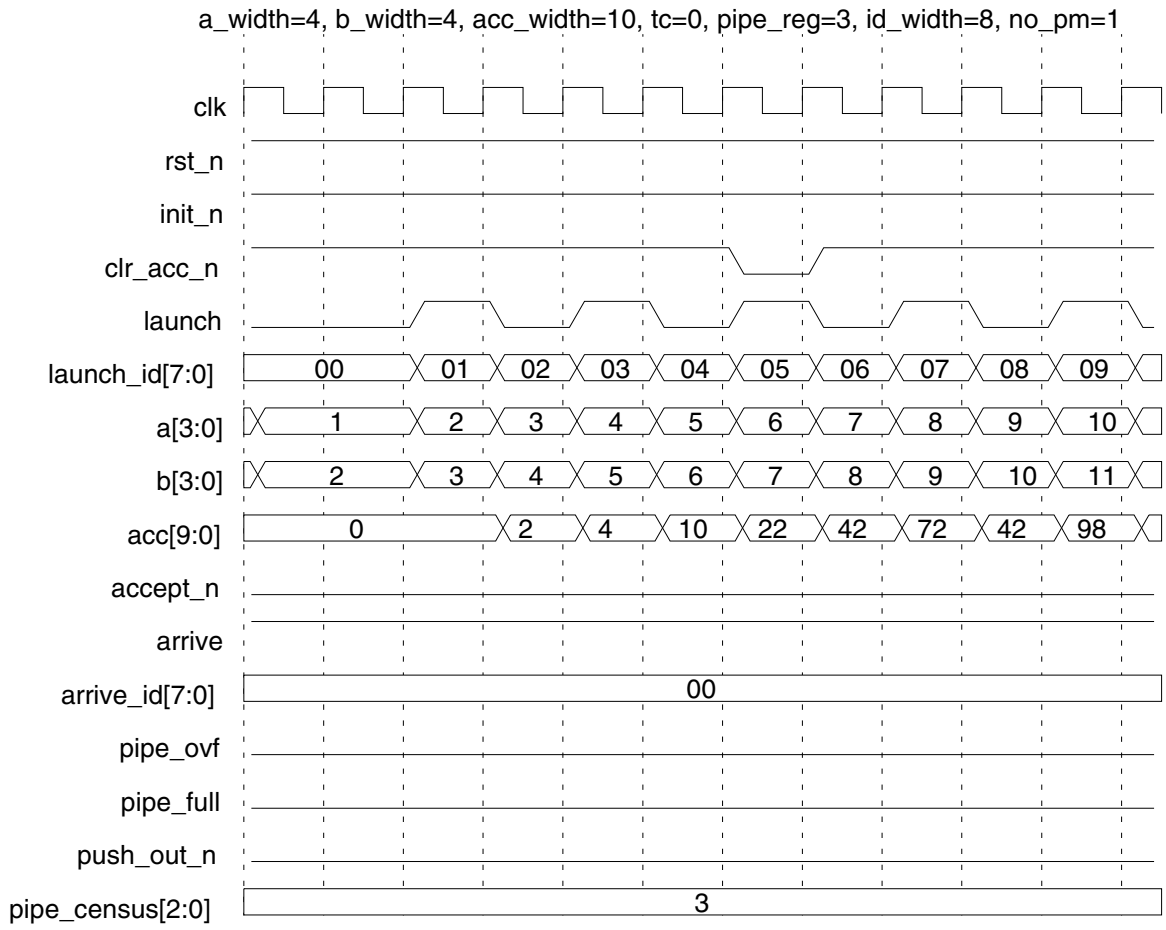
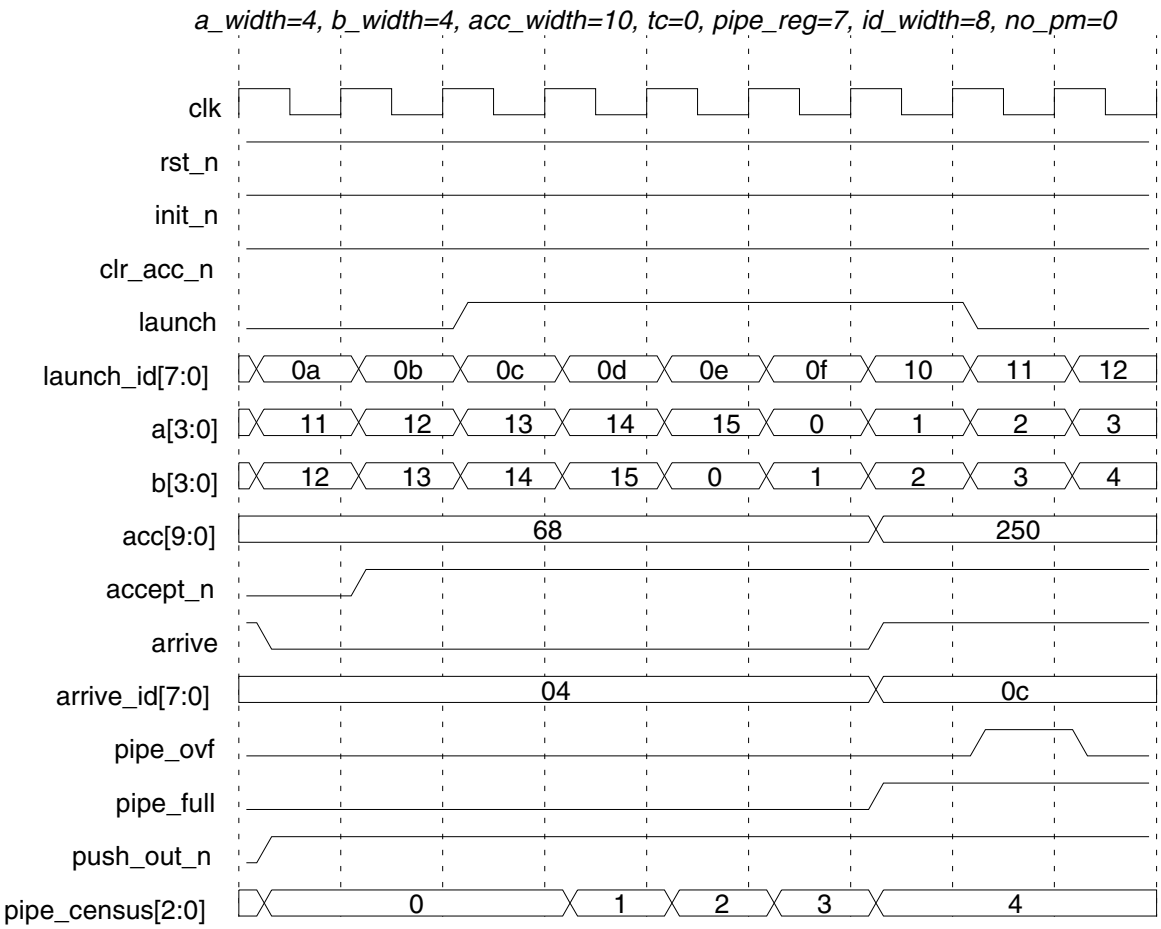


Figure 1-6 depicts the condition in which a pipeline overflow occurs. When `launch` is asserted during the time `pipe_full` is 1 and `accept_n` is 1 (de-asserted), the `pipe_ovf` signal goes active on the next clock cycle since it is registered.

Figure 1-6 Pipeline Overflow



Enabling minPower

You can instantiate this component without enabling minPower, but to achieve datapath gating power savings, you must enable minPower optimization, as follows:

- Design Compiler
  - Version P-2019.03 and later:

    ```
    set power_enable_minpower true
    ```

  - Before version P-2019.03 (requires the DesignWare-LP license feature):

    ```
    set synthetic_library {dw_foundation.sldb dw_minpower.sldb}
    set link_library {* $target_library $synthetic_library}
    ```

- Fusion Compiler

  Optimization for minPower is enabled as part of the total_power metric setting. To enable the total_power metric, use the following:

  ```
  set_qor_strategy -stage synthesis -metric total_power
  ```

Related Topics

- Math – Arithmetic Overview
- DesignWare Building Block IP User Guide

HDL Usage Through Component Instantiation - VHDL

```
library IEEE,DWARE;
use IEEE.std_logic_1164.all;
use DWARE.DW_Foundation_comp.all;

entity DW_piped_mac_inst is
      generic (
        inst_a_width : POSITIVE := 8;
        inst_b_width : POSITIVE := 8;
        inst_acc_width : POSITIVE := 16;
        inst_tc : NATURAL := 0;
        inst_pipe_reg : NATURAL := 0;
        inst_id_width : POSITIVE := 1;
        inst_no_pm : NATURAL := 0;
        inst_op_iso_mode : NATURAL := 0
        );
      port (
        inst_clk : in std_logic;
        inst_rst_n : in std_logic;
        inst_init_n : in std_logic;
        inst_clr_acc_n : in std_logic;
        inst_a : in std_logic_vector(inst_a_width-1 downto 0);
        inst_b : in std_logic_vector(inst_b_width-1 downto 0);
        acc_inst : out std_logic_vector(inst_acc_width-1 downto 0);
        inst_launch : in std_logic;
        inst_launch_id : in std_logic_vector(inst_id_width-1 downto 0);
        pipe_full_inst : out std_logic;
        pipe_ovf_inst : out std_logic;
        inst_accept_n : in std_logic;
        arrive_inst : out std_logic;
        arrive_id_inst : out std_logic_vector(inst_id_width-1 downto 0);
        push_out_n_inst : out std_logic;
        pipe_census_inst : out std_logic_vector(2 downto 0)
        );
    end DW_piped_mac_inst;


architecture inst of DW_piped_mac_inst is

begin

    -- Instance of DW_piped_mac
    U1 : DW_piped_mac
    generic map ( a_width => inst_a_width,
                  b_width => inst_b_width,
                  acc_width => inst_acc_width,
                  tc => inst_tc,
                  pipe_reg => inst_pipe_reg,
```

```
                id_width => inst_id_width,
                no_pm => inst_no_pm,
                op_iso_mode => inst_op_iso_mode )

    port map ( clk => inst_clk,
                   rst_n => inst_rst_n,
                   init_n => inst_init_n,
                   clr_acc_n => inst_clr_acc_n,
                   a => inst_a,
                   b => inst_b,
                   acc => acc_inst,
                   launch => inst_launch,
                   launch_id => inst_launch_id,
                   pipe_full => pipe_full_inst,
                   pipe_ovf => pipe_ovf_inst,
                   accept_n => inst_accept_n,
                   arrive => arrive_inst,
                   arrive_id => arrive_id_inst,
                   push_out_n => push_out_n_inst,
                   pipe_census => pipe_census_inst );
end inst;

-- Configuration for use with a VHDL simulator
-- pragma translate_off
library DW03;
configuration DW_piped_mac_inst_cfg_inst of DW_piped_mac_inst is
  for inst
  end for; -- inst
end DW_piped_mac_inst_cfg_inst;
-- pragma translate_on
```

HDL Usage Through Component Instantiation - Verilog

```
module DW_piped_mac_inst( inst_clk, inst_rst_n, inst_init_n,
    inst_clr_acc_n, inst_a, inst_b, acc_inst, inst_launch, inst_launch_id,
    pipe_full_inst, pipe_ovf_inst, inst_accept_n, arrive_inst,
    arrive_id_inst, push_out_n_inst, pipe_census_inst );

parameter inst_a_width = 8;
parameter inst_b_width = 8;
parameter inst_acc_width = 16;
parameter inst_tc = 0;
parameter inst_pipe_reg = 0;
parameter inst_id_width = 1;
parameter inst_no_pm = 0;
parameter inst_op_iso_mode = 0;

input inst_clk;
input inst_rst_n;
input inst_init_n;
input inst_clr_acc_n;
input [inst_a_width-1 : 0] inst_a;
input [inst_b_width-1 : 0] inst_b;
output [inst_acc_width-1 : 0] acc_inst;
input inst_launch;
input [inst_id_width-1 : 0] inst_launch_id;
output pipe_full_inst;
output pipe_ovf_inst;
input inst_accept_n;
output arrive_inst;
output [inst_id_width-1 : 0] arrive_id_inst;
output push_out_n_inst;
output [2 : 0] pipe_census_inst;

 // Instance of DW_piped_mac
 DW_piped_mac #(inst_a_width, inst_b_width, inst_acc_width,
          inst_tc, inst_pipe_reg, inst_id_width, inst_no_pm)
      U1 ( .clk(inst_clk),
                .rst_n(inst_rst_n),
                .init_n(inst_init_n),
                .clr_acc_n(inst_clr_acc_n),
                .a(inst_a),
                .b(inst_b),
                .acc(acc_inst),
                .launch(inst_launch),
                .launch_id(inst_launch_id),
                .pipe_full(pipe_full_inst),
                .pipe_ovf(pipe_ovf_inst),
                .accept_n(inst_accept_n),
                .arrive(arrive_inst),
```

```
                .arrive_id(arrive_id_inst),
                .push_out_n(push_out_n_inst),
                .pipe_census(pipe_census_inst) );

  endmodule
```

Revision History

For notes about this release, see the *DesignWare Building Block IP Release Notes*.

For lists of both known and fixed issues for this component, refer to the STAR report.

For a version of this datasheet with visible change bars, click here.

Date	Release	Updates
July 2020	DWBB_201912.5	■ Adjusted content and title of "Suppressing Warning Messages During Verilog Simulation" on page 7 and added the DW_SUPPRESS_WARN macro
October 2019	DWBB_201903.5	■ Added the "Disabling Clock Monitor Messages" section
March 2019	DWBB_201903.0	■ Clarified the op_iso_mode parameter in Table 1-2 on page 2 ■ Clarified license requirements in Table 1-3 on page 3 ■ Added "Enabling minPower" on page 13 ■ Added this Revision History table and the document links on this page

Copyright Notice and Proprietary Information

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html.

All other product or company names may be trademarks of their respective owners.

Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

Synopsys, Inc.
www.synopsys.com